Los Alamos NATIONAL LABORATORY EST.1943

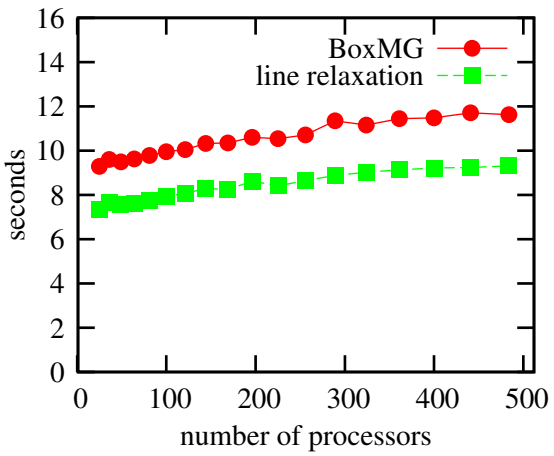
An Efficient, Numerically Stable, and Scalable Parallel Tridiagonal Solver

Christian Ketelsen, christian.ketelsen@colorado.edu
Markus Berndt, berndt@lanl.gov
J. David Moulton, moulton@lanl.gov

Introduction

We describe a stable, efficient, parallel algorithm for the solution of diagonally dominant tridiagonal linear systems that scales well on distributed memory parallel computers. This algorithm is in the class of partitioning algorithms. Its multi-level recursive design makes it well suited for distributed memory parallel computers with very large numbers of processors.

The need to solve large tridiagonal systems arises in many numerical analysis applications. In our work, they arise in line relaxations needed by robust multigrid methods, such as the parallel BoxMG code [1], for structured grid problems. The following figure illustrates the importance of having efficient line relaxation methods as the majority of time in a multigrid solve is spent performing these relaxation steps.



Timings for 20 $V(1,1)$ BoxMG cycles with red-black line relaxation on square processor grids ranging from 2×2 to 22×22.

Background

On a serial computer, Gaussian elimination without pivoting can be used to solve a diagonally dominant tridiagonal system of linear equations in $O(N)$ steps. This algorithm is commonly referred to as the Thomas algorithm[1]. Unfortunately, this algorithm is not well suited for parallel computers. The first parallel algorithm for the solution of tridiagonal systems was developed by Hockney and Golub. It is now usually referred to as cyclic reduction. Stone introduced his recursive doubling algorithm in 1973. Both cyclic reduction and recursive doubling are designed for fine grained parallelism, where each processor owns exactly one row of the tridiagonal matrix. In 1981, Wang proposed a partitioning algorithm aimed at more coarse-grained parallel computation typical for shared memory clusters, where $N_P << N$. There has also been attention directed toward a parallel partitioning of the standard LU algorithm. In 1986, Sun et al. introduced the parallel partitioning LU algorithm that is very similar to Bondeli's divide and conquer algorithm. These algorithms, while well suited for problems distributed across a moderately large number of processors, do not scale well to very large numbers of pocessors.

[1]The references to the papers that we referred to in this section can be found in [2].

Algorithm

Our algorithm can be described as a recursion with a partitioning algorithm as its basis. We begin by describing this partitioning algorithm. The tridiagonal linear system is assumed to be distributed across a large number of processors, such that each processor owns a contiguous number of rows. Each processor transforms its piece of the tridiagonal matrix into a matrix with a sparsity

pattern such as

$$A_{local} = \begin{pmatrix} \times & \times & & & & \times & \\ & \times & \times & \times & & & \\ & & \times & \times & \times & & \\ & & & \times & \times & \times & \\ & \times & & & & \times & \times \end{pmatrix}.$$

The first and last rows (red) from each processor are what we term interface equations because they depend only on the unknowns located on the boundary of the current processor and its nearest neighbors. These interface equations are computed by taking a linear combination of local rows on each processor. The weights used in this combination are found by solving a smaller tridiagonal sytem with the blue ✖'s as its coefficient matrix; we denote this sub-matrix by $\hat{A}_{local}$. In practice this is done using the Cholesky decomposition of the symmetric positive definite matrix $\hat{A}_{local} = LL^t$, which is stored and reused at a later stage of the algorithm.

Gathering the interface equations from every processor yields an interface system that is again tridiagonal and diagonally dominant. This interface system can be solved by gathering all equations to one processor, using the Thomas algorithm there, and then scattering the solution back to all N_P processors. Then the interface unknowns can be eliminated from the local systems, yielding N_P local tridiagonal systems (blue ✖'s). These local systems can be solved efficiently and in a numerically stable manner using the same Cholesky decomposition that was computed in the first phase of the algorithm.

Although this non-recursive single-level approach scales reasonably well for moderate numbers of processors, it does not scale well for very large number of processors: Gather and scatter operations typically scale linearly with the number of of processors. Our remedy is to divide the set of all processors into groups that own contiguous sets of unknowns. Each such group gathers its interface equations to one processor in the group (the head processor). The head processor then uses these interface equations to compute a new pair of lower level interface equations. This yields a lower level interface system on the subset of all head processors. We proceed with further recursion if the subset of processors is sufficiently large, or solve the new interface system directly on one of the processors in the subset.

The size of the processor groups that is used in the coarsening step is hardware dependent and the best value for a given parallel computer should be determined a-priori by numerical experiment.

Acknowledgements

Los Alamos Report LA-UR-06-XXXX. This work was carried out under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396, and supported by the Advanced Simulation and Computing Program.

References

[1] T. M. Austin, M. Berndt, B. K. Bergen, J. E. Dendy, and J. D. Moulton. Parallel, scalable, and robust multigrid on structured grids. Technical report, Los Alamos National Laboratory, LA-UR-03-9167, 2003.

[2] T. M. Austin, M. Berndt, and J. D. Moulton. A memory efficient parallel line solver. Technical report, 2004. LA-UR-04-4149.